Analysis

December 2, 2022

[ ]:
```
import pandas as pd
import datetime
import numpy as np
```

[ ]:
```
df = pd.read_csv('output.csv', index_col=0)
```

[ ]:
```
df['Transaction Date and Time'] = pd.to_datetime(df['Transaction Date and␣
↪Time'])
df.info()
```

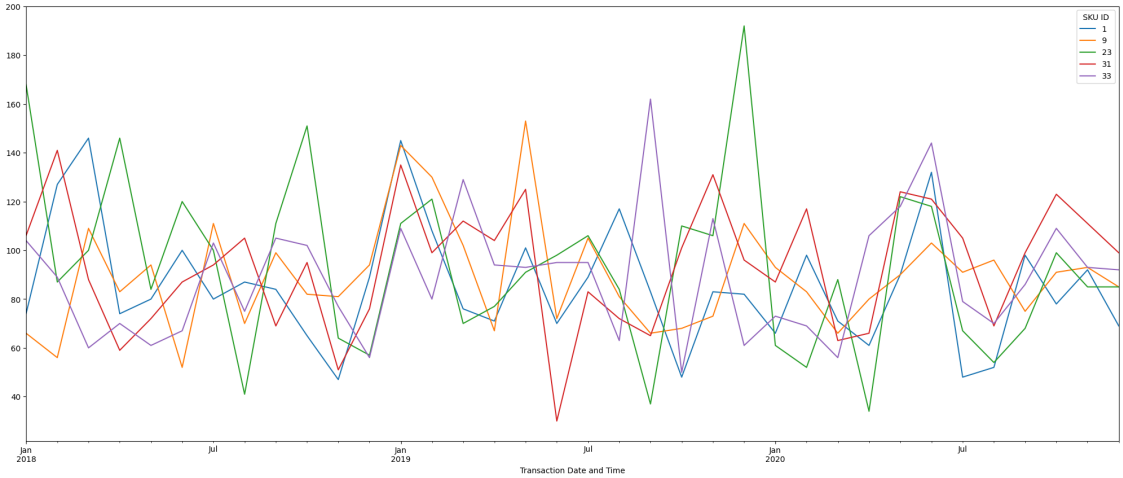
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 97805 entries, 0 to 97804
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Customer Name              97805 non-null  object
 1   Transaction Date and Time  97805 non-null  datetime64[ns]
 2   Customer Age               97805 non-null  int64
 3   Location                   97805 non-null  object
 4   Discount                   97805 non-null  float64
 5   SKU ID                     97805 non-null  int64
 6   Quantity                   97805 non-null  int64
 7   Price                      97805 non-null  float64
 8   Name                       97805 non-null  object
 9   Description                97805 non-null  object
 10  Amount Spent in USD        97805 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(3), object(4)
memory usage: 9.0+ MB
```

[ ]:
```
#time series from 2018-2020, top 5 items sold by quantity, daily transaction␣
↪sold
```

[ ]:
```
top_5_sold = list(df.groupby('SKU ID')['Quantity'].sum().sort_values().iloc[0:
↪5].index)
window = df[(df['Transaction Date and Time']>='01-01-2018') & (df['Transaction␣
↪Date and Time']<='12-31-2020') & (df['SKU ID'].isin(top_5_sold))]
```

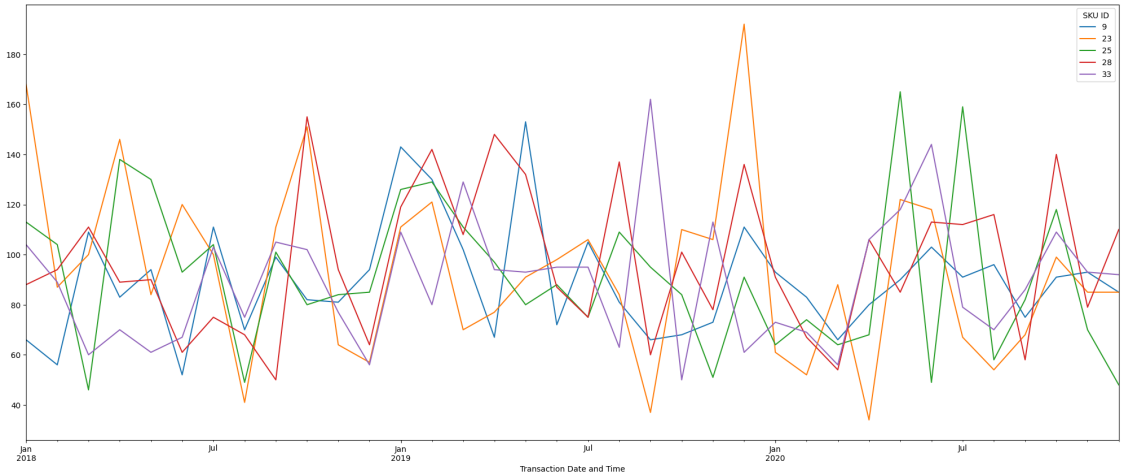
```
pd.pivot_table(window, values='Quantity', index="Transaction Date and Time",␣
↪columns=['SKU ID'], aggfunc=np.sum).resample('M').sum().plot(kind='line',␣
↪figsize=(25,10))
```

[ ]: <AxesSubplot: xlabel='Transaction Date and Time'>



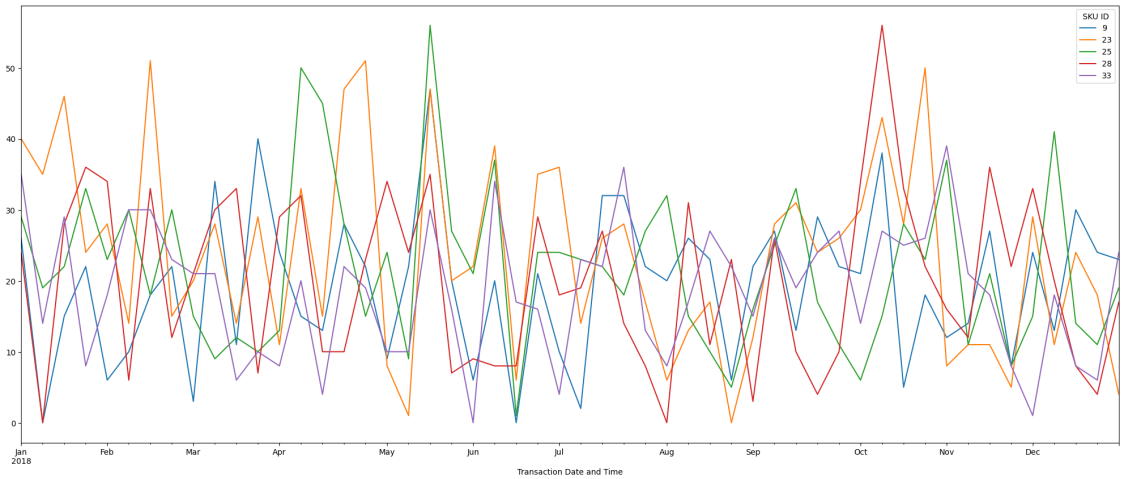
[ ]:
```
# repeat for top 5 items by revenue
top_5_rev = list(df.groupby('SKU ID')['Amount Spent in USD'].sum().
↪sort_values().iloc[0:5].index)
window = df[(df['Transaction Date and Time']>='01-01-2018') & (df['Transaction␣
↪Date and Time']<='12-31-2020') & (df['SKU ID'].isin(top_5_rev))]
pd.pivot_table(window, values='Amount Spent in USD', index="Transaction Date␣
↪and Time", columns=['SKU ID'], aggfunc=np.sum).resample('M').sum().
↪plot(kind='line', figsize=(25,10))
```

[ ]: <AxesSubplot: xlabel='Transaction Date and Time'>



[ ]:
```
#1st plot is 2018 of top 5 products by revenue
window = df[(df['Transaction Date and Time']>='01-01-2018') & (df['Transaction␣
↪Date and Time']<='12-31-2018') & (df['SKU ID'].isin(top_5_rev))]
pd.pivot_table(window, values='Amount Spent in USD', index="Transaction Date␣
↪and Time", columns=['SKU ID'], aggfunc=np.sum).resample('W').sum().
↪plot(kind='line', figsize=(25,10))
```

[ ]:
```
<AxesSubplot: xlabel='Transaction Date and Time'>
```



[ ]:
```
#2nd plot of 2018 and 2019, bar graph, identify volatile product in terms of␣
↪sales per week
# identify within this period the most volatile in terms of gross sales per␣
↪week
window = df[(df['Transaction Date and Time']>='01-01-2018') & (df['Transaction␣
↪Date and Time']<='12-31-2019')]
window.groupby('SKU ID').resample('W', on='Transaction Date and Time').
↪std(numeric_only=True).sort_values(by='Quantity', ascending=False)[0:20]
```

[ ]:

SKU ID	Transaction Date and Time	Customer Age	Discount	SKU ID	Quantity	\
33	2019-03-03	12.727922	0.0	0.0	6.363961	
3	2018-05-06	7.071068	0.0	0.0	6.363961	
11	2019-10-06	11.313708	0.0	0.0	6.363961	
17	2019-03-10	0.000000	0.0	0.0	6.363961	
29	2018-01-14	6.363961	0.0	0.0	6.363961	
6	2018-04-29	3.535534	0.0	0.0	6.363961	
7	2018-05-13	37.476659	0.0	0.0	6.363961	
8	2018-03-18	2.828427	0.0	0.0	6.363961	

28	2018-08-19	19.798990	0.0	0.0	6.363961
23	2018-12-09	2.828427	0.0	0.0	6.363961
13	2018-09-09	9.899495	0.0	0.0	6.363961
31	2019-11-03	14.849242	0.0	0.0	6.363961
15	2018-06-24	15.556349	0.0	0.0	6.363961
14	2019-04-28	25.455844	0.0	0.0	6.363961
13	2019-04-07	20.506097	0.0	0.0	5.656854
26	2019-07-07	0.707107	0.0	0.0	5.656854
28	2018-02-25	17.677670	0.0	0.0	5.656854
12	2018-06-17	29.698485	0.0	0.0	5.656854
17	2018-09-02	8.485281	0.0	0.0	5.656854
29	2018-05-13	10.606602	0.0	0.0	5.656854

		Price	Amount Spent in USD
SKU ID	Transaction Date and Time		
33	2019-03-03	0.0	6.363961
3	2018-05-06	0.0	19.091883
11	2019-10-06	0.0	25.455844
17	2019-03-10	0.0	38.183766
29	2018-01-14	0.0	25.455844
6	2018-04-29	0.0	19.091883
7	2018-05-13	0.0	12.727922
8	2018-03-18	0.0	12.727922
28	2018-08-19	0.0	6.363961
23	2018-12-09	0.0	6.363961
13	2018-09-09	0.0	25.455844
31	2019-11-03	0.0	12.727922
15	2018-06-24	0.0	25.455844
14	2019-04-28	0.0	31.819805
13	2019-04-07	0.0	22.627417
26	2019-07-07	0.0	5.656854
28	2018-02-25	0.0	5.656854
12	2018-06-17	0.0	22.627417
17	2018-09-02	0.0	33.941125
29	2018-05-13	0.0	22.627417

[ ]:
```
#3rd plot is comparing 2018,2019,2020 histograms
# histogram 2018, 2019, 2020 most volatile
```

[ ]:
```
# 3 new raw data features: discount, weight, volume
# 3 new composite indexes that make intuitive
# graphs look pretty
# EDA - statistical signifance
```

[ ]:
```
#graph total sales on weekly basis
df.resample('W', on='Transaction Date and Time').sum()['Quantity'].
 ↪plot(kind='line', figsize=(25,10))
```

```
/tmp/ipykernel_3274/4263859810.py:2: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.sum is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df.resample('W', on='Transaction Date and
Time').sum()['Quantity'].plot(kind='line', figsize=(25,10))
```

[ ]: <AxesSubplot: xlabel='Transaction Date and Time'>